Henry Stewart

mparison points between UCSD Pascal and ATARI Pascal
epared by MT MicroSYSTEMS
cember 15, 1980

c ities available in UCSD Pascal not found in ATARI Pascal

Segment procedures
(This facility allows overlays from disk but costs in execution time)

Long Integers

Units for modular compilation
(Clean and clear but restricted)

Bit level packing on PACKED structures
(costs in interpreter code plus multiply and divide operations in accessing fields)

UCSD Pascal Operating System Dependent features
UNIT I/O (similar to XIO on ATARI)
Long file names

EXIT with procedure name
(difficult to implement in Native code environment without undue overhead)

Type INTERACTIVE
(ATARI Pascal files which are associated with the console are automatically interactive files, no special declaration is required)

SEEK procedure
(ATARI Pascal has SEEKREAD and SEEKWRITE which are not so confusing (in UCSD you must GET or PUT after a SEEK and if not then things become very confused; on ATARI SEEKREAD and SEEKWRITE contain implied GET/PUT logic))

Sets More flexible
Sets in UCSD Pascal may be longer (up to 4090 bits) but are significantly slower than ATARI Pascal fixed size sets

Pascal oriented screen editor

---

atures available in ATARI Pascal not found in UCSD Pascal

More flexible modular compilation
Local Static variables, external procedures and functions located in main program, external global variable usage all are not available in UCSD

ATARI Operating System compatibility
Pascal and BASIC live together on the same disk
File interchange is possible
User does not need to learn two operating systems

Pascal interfaces to I/O similar to basic:

- XIO
- Graphics
- Sound
- Paddle/Joystick I/O
- Named I/O devices (exactly the same as BASIC)

Complete ISO standard Pascal

- Conformant Array handling
- Procedures and Functions can be passed as parameters
- GOTO out of a procedure into a surrounding procedure allowed
- ATARI Pascal passes all validation suite (UCSD Fails many tests)

Native 6502 code option

Faster P-code
ATARI P-code programs run up to 2 times faster than similar programs on the APPLE-II

Interchangable P-code/Native code modules
Modules may be either P-code, Native code Pascal or assembly language as desired. All routines use the same calling conventions and no special syntax required for assembly language external procedures as is required in UCSD Pascal

Built-in BYTE data type
Eliminates the use of confusing CASE variant records to manipulate characters as integers

Built-in WORD data type
An unsigned 16-bit data type very useful for address arithmetic and machine level programing

Facilities for user interception of errors (@ERR routine)
User can catch and therefore not allow program to abort divide by zero, string truncation, range errors and heap overflow.

I/O protection built-in
In UCSD Pascal when a program reads a string if the string is too long for the receiving variable the I/O code simply overwrites the bytes following the string in memory. ATARI Pascal truncates the input to the proper length and does not overwrite any other data in memory

Better character/string compatibility
UCSD Pascal did not fully implement compatibility between strings and characters. ATARI Pascal does.

Relaxable type-checking
For applications which are system dependent ATARI Pascal allows relaxation of type checking to allow machine I/O and memory manipulation to be done without cluttering the program with confusing CASE variant records. If a program is non-portable then why make it unreadable?

Full dynamic heap management (NEW and DISPOSE)
ATARI Pascal fully implements the NEW and DISPOSE procedures including fragmentation management and re-use of disposed areas. UCSD Pascal only implements a stack oriented heap which is significantly less flexible

Higher precision reals

ATARI Pascal has 8-10 digits of precision on real numbers
UCSD has only 6.5 digits

5. Temporary files
ATARI Pascal totally implements local files as specified
in the standard. (UCSD does not implement this feature
at all)

6. Files allowed in procedures / records / arrays
ATARI Pascal fully implements files in all legal areas.
UCSD does not allow local files, files in records or
arrays of files.

7. ADDR function
ATARI Pascal has a function which returns the address
of a variable or a procedure/function which is useful
when doing machine dependent programming

8. Built-in portable bit-manipulation routines
In UCSD Pascal bit-manipulation is done using CASE
variant records (a very unclear, unporatble method)
ATARI Pascal contans TSTBIT, SETBIT, CLRBIT, SHL and
SHR routines

9. High speed file I/O routines
In both ATARI Pascal and UCSD Pascal the GET/PUT file
I/O is notoriously slow. ATARI Pascal also contains
GNB/WNB a set of high-speed I/O routines for byte file
input/output

PACK/UNPACK fully implemented
UCSD does not implement the PACK and UNPACK procedures
which are necessary for portable programs using the
ISO standard

Compile time constants
ATARI Pascal has a built-in INLINE feature which can
be used to generate compile-time constant data which
removes the need for code to initialize large constant
tables

Read/Write on non-text files
UCSD does not implement READ and WRITE for non-text
files as specified in the ISO standard

Boolean output
UCSD does not implement WRITE/WRITELN of boolean
expressions as specified by the standard

Non-decimal output
ATARI Pascal has facilities for output in any base
from 2 through 16 (2,8,10 and 16 being the most useful)

Non-decimal input
ATARI Pascal supports input of either decimal or hex numbers

Program chaining
UCSD segment procedures are limited to 6 per program this
limits the development of large applications which typicaly
make take 10 to 50 overlays. Therefore large programs can
be developed in ATARI Pascal

Standard RESET/REWRITE file parameters

ATARI Pascal has not extended the parameter list on any
ISO standard routine. For accessing external files
a new procedure (ASSIGN) has been added to associate
an external file name with a file variable

ELSE on CASE statement
In UCSD Pascal users must typically compare using a set
expression before executing a CASE statement to see if the
selecting expression will result in at least one statement
being executed. In ATARI Pascal the ELSE clause allows
selecting expressions which do not match a selector to
be handled in a clean easy to read manner

Faster sets
While UCSD sets may be larger the small, statically sized
ATARI Pascal sets can be significantly faster than UCSD
Pascal sets.